TILBURG UNIVERSITY

Tilburg University

An Analytic Center Cutting Plane Method to Determine Complete Positivity of a Matrix

Badenbroek, Riley; de Klerk, Etienne

Publication date:
2020

Document Version
Early version, also known as pre-print

Link to publication in Tilburg University Research Portal

Citation for published version (APA):
Badenbroek, R., & de Klerk, E. (2020). *An Analytic Center Cutting Plane Method to Determine Complete Positivity of a Matrix*. (arXiv; Vol. 2006.05319). Cornell University Library. https://arxiv.org/abs/2006.05319

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 05. May. 2023

arXiv:2006.05319v1 [math.OC] 9 Jun 2020

An Analytic Center Cutting Plane Method to Determine Complete Positivity of a Matrix

Riley Badenbroek Etienne de Klerk

June 11, 2020

Abstract

We propose an analytic center cutting plane method to determine if a matrix is completely positive, and return a cut that separates it from the completely positive cone if not. This was stated as an open (computational) problem by Berman, Dür, and Shaked-Monderer [Electronic Journal of Linear Algebra, 2015]. Our method optimizes over the intersection of a ball and the copositive cone, where membership is determined by solving a mixed-integer linear program suggested by Xia, Vera, and Zuluaga [INFORMS Journal on Computing, 2018]. Thus, our algorithm can, more generally, be used to solve any copositive optimization problem, provided one knows the radius of a ball containing an optimal solution. Numerical experiments show that the number of oracle calls (matrix copositivity checks) for our implementation scales well with the matrix size, growing roughly like $O(d^2)$ for $d \times d$ matrices. The method is implemented in Julia, and available at `https://github.com/rileybadenbroek/CopositiveAnalyticCenter.jl`.

Keywords: copositive optimization, analytic center cutting plane method, completely positive matrices
AMS subject classification: 90C25, 90C51, 49M05, 65K05

1 Introduction

We define the completely positive cone $\mathcal{CP}^d \subset \mathbb{S}^d$ as

$$\mathcal{CP}^d := \{BB^\top : B \geq 0, B \in \mathbb{R}^{d\times k} \text{ for some } k\},$$

where $\mathbb{S}^d$ denotes the space of real symmetric $d \times d$ matrices. Completely positive matrices play an important role in optimization. For instance, by a theorem of Motzkin and Straus [30] (see also De Klerk and Pasechnik [13]), the stability number of a graph can be formulated as an optimization problem with linear objective and linear constraints over the completely positive cone (or its dual cone). A seminal result by Burer [11] shows that – under mild assumptions – binary quadratic problems can also be reformulated as optimization problems over the completely positive cone. Other applications build on the work by Kemperman and Skibinsky [26], who found that

$$\left\{ \int xx^\top \, \mathrm{d}\mu(x) : \mu \text{ is a finite-valued nonnegative measure supported on } \mathbb{R}^d_+ \right\} = \mathcal{CP}^d.$$

This equality has spawned a large number of applications in distributionally robust optimization, e.g. Natarajan et al. [32] and Kong et al. [27] (see Li et al. [29] for a survey).

One advantage of these reformulations is that they transform hard problems into linear optimization problems over a proper cone, which allow them to benefit from the (duality) theory of convex optimization. The difficulty in such problems is essentially moved to the conic constraint. It is therefore unsurprising that even testing whether a matrix is completely positive is NP-hard, cf. Dickinson and Gijben [16]. Several approaches to this testing problem exist in the literature.

Jarre and Schmallowsky [25] propose an augmented primal-dual method that provides a certificate if $C \in \mathcal{CP}^d$ by solving a sequence of second-order cone problems. However, their algorithm converges slowly if C is on the boundary of $\mathcal{CP}^d$, and the regularization they propose to solve this is computationally expensive.

An obvious way to verify that C is completely positive is to find a factorization $C = BB^\top$ where $B \geq 0$. Several authors have done this for specific matrix structures, see Dickinson and Dür [15], Bomze [5], and the references therein. For general matrices, factorization methods have been proposed by Nie [33], and Sponsel and Dür [39], but these methods do not perform well on bigger matrices. Groetzner and Dür [23] develop an alternating projection scheme that does scale well, but is not guaranteed to find a factorization for a given completely positive matrix. The method struggles in particular for matrices near the boundary of the completely positive cone. Another heuristic method based on projection is given by Elser [17]. Sikirić et al. [38] can find a rational factorization whenever it exists, although the running time is hard to predict.

To actually optimize over the completely positive cone is even harder. Bomze et al. [7] suggest a factorization heuristic with promising numerical performance. A more naive approach to solving completely positive optimization problems is to replace the cone $\mathcal{CP}^d$ with a tractable outer approximation, such as the cone of doubly nonnegative matrices (i.e. the symmetric positive semidefinite matrices with nonnegative elements). If the problem over this outer approximation has an optimal solution C, one would not only like to check if $C \in \mathcal{CP}^d$, but also to generate a cut that separates C from $\mathcal{CP}^d$ if $C \notin \mathcal{CP}^d$. After adding the cut to the relaxation, the relaxation may be re-solved, hopefully yielding a better solution (this scheme is mentioned in e.g. Sponsel and Dür [39] and Berman, Dür, and Shaked-Monderer [4]).

Burer and Dong [12] proposed a method to generate such a cut for 5×5 matrices. Sponsel and Dür [39] suggested an algorithm based on simplicial partition. Nevertheless, finding a cutting plane for the completely positive matrices is still listed as an open problem by Berman, Dür, and Shaked-Monderer [4].

Our approach will optimize over the dual cone of $\mathcal{CP}^d$ (with respect to the trace inner product $\langle \cdot, \cdot \rangle$), which is known as the copositive cone. This cone is defined as

$$\mathcal{COP}^d := \{X \in \mathbb{S}^d : y^\top X y \geq 0 \text{ for all } y \in \mathbb{R}^d_+\}.$$

It is well known that $C \in \mathcal{CP}^d$ if and only if $\langle C, X \rangle \geq 0$ for all $X \in \mathcal{COP}^d$. Hence, minimizing $\langle C, X \rangle$ over $X \in \mathcal{COP}^d$ should give us an answer to the question if $C \in \mathcal{CP}^d$ or not, and if not, we immediately have an $X \in \mathcal{COP}^d$ that induces a valid cut.

It should be noted that determining if a matrix X lies in $\mathcal{COP}^d$ is co-NP-complete, see Murty and Kabadi [31]. The classical copositivity test is due to Gaddum [19], but his procedure requires performing a test for all principal minors of a matrix, which does not scale well to larger d. Nie et al. [34] have proposed an algorithm based on semidefinite programming that terminates in finite time, although the actual computation time is hard to predict. Anstreicher [1] shows that copositivity can be tested by solving a mixed-integer linear program (MILP), building on work by Dickinson [14]. See Hiriart-Urruty and Seeger [24] for a review of the properties of copositive matrices.

Our chosen method of testing if a matrix X is copositive is the same as in Badenbroek and De Klerk [3], which is similar to Anstreicher's. Our method also solves an MILP, and also admits a $y \geq 0$ such that $y^\top X y < 0$ if X is not copositive. The main difference is that our method derives from Xia et al. [40] instead of Dickinson [14].

Since the copositive cone is intractable, it will have to be replaced by an approximation if we want to optimize over it. Bundfuss and Dür [9, 10] use polyhedral inner and outer approximations based on simplicial partitions that are refined in regions interesting to the optimization. Hierarchies of inner approximations of the copositive cone are proposed by Parrilo [35], De Klerk and Pasechnik [13] (see also Bomze and De Klerk [6]) and Peña et al. [36]. Yıldırım [42] proposes polyhedral outer approximations of the copositive cone, and analyzes the gap to the inner approximations by De Klerk and Pasechnik. Finally, Lasserre [28] proposes a spectrahedral hierarchy of outer approximations of $\mathcal{COP}^d$.

Our approach to optimize over the copositive cone is to use an analytic center cutting plane method. Therefore, it is convenient to use a simple polyhedral outer approximation of the copositive cone: $\{X \in \mathbb{S}^d : y^\top X y \geq 0 \, \forall y \in \mathcal{Y}\}$, where $\mathcal{Y} \subset \mathbb{R}^d_+$ is a finite set of vectors. These vectors will be generated by performing

the copositivity check for some matrix X, and if it turns out there exists a $y \geq 0$ such that $y^\top X y < 0$, this y is added to $\mathcal{Y}$.

Analytic center cutting plane methods were first introduced by Goffin and Vial [21] (see [22] for a survey by the same authors, or Boyd et al. [8]). The advantage of analytic center cutting plane methods is that the number of iterations scales reasonably with the problem dimension. For instance, Goffin et al. [20] find that the number of iterations is $O^*(n^2/\epsilon^2)$, where n is the number of variables, ϵ is the desired accuracy, and O^* ignores polylogarithmic terms. In every iteration of our algorithm, the main computational effort is solving an MILP whose size does not change throughout the algorithm's run.

We describe our method in detail Section 2 and conduct numerical experiments in Section 3.

Notation

Throughout this work, we use the Euclidean inner product on $\mathbb{R}^n$, and the trace inner product $\langle \cdot, \cdot \rangle$ on $\mathbb{S}^d$. For some vector $x \in \mathbb{R}^n$ with all elements unequal to 0, and some integer $i \in \mathbb{Z}$, let $x^i \coloneqq \begin{bmatrix} x_1^i & \cdots & x_n^i \end{bmatrix}^\top$.

Since $\mathbb{S}^d$ is isomorphic to $\mathbb{R}^{d(d+1)/2}$, we can also consider our optimization over $\mathcal{COP}^d$ as an optimization over $\mathbb{R}^{d(d+1)/2}$. To do that, we follow the convention from Julia's `MathOptInterface` package, which (implicitly) uses the following vectorization operator on $X = [X_{ij}] \in \mathbb{S}^d$:

$$\operatorname{vec}(X) \coloneqq \begin{bmatrix} X_{11} & X_{12} & X_{22} & X_{13} & X_{23} & \cdots & X_{dd} \end{bmatrix}^\top,$$

i.e. $\operatorname{vec}(X)$ contains the upper triangular part of the matrix. Let $\operatorname{mat} : \mathbb{R}^{d(d+1)/2} \to \mathbb{S}^d$ be the inverse of vec, and let $\operatorname{mat}^* : \mathbb{S}^d \to \mathbb{R}^{d(d+1)/2}$ be the adjoint of mat.

The problem we will look at for some fixed $C \in \mathbb{S}^d$ is

$$\min_X \{ \langle C, X \rangle : \|\operatorname{vec}(X)\|^2 \leq 1, X \in \mathcal{COP}^d \}. \tag{1}$$

If $X \in \mathcal{COP}^d$ is an optimal solution to (1), then $C \in \mathcal{CP}^d$ if and only if $\langle C, X \rangle \geq 0$.

2 An Analytic Center Cutting Plane Method

Analytic center cutting plane methods can be used to solve optimization problems of the form

$$\inf_x \{ c^\top x : x \in \mathcal{X} \subseteq \mathbb{R}^n \}, \tag{2}$$

where $c \in \mathbb{R}^n$ and $\mathcal{X}$ is a nonempty, bounded, convex set for which we know a separation oracle. In other words, given some point $x \in \mathbb{R}^n$, one should be able to determine if $x \in \mathcal{X}$ or not, and moreover, if $x \notin \mathcal{X}$, we must be able to generate a halfspace $\mathcal{H}$ such that $\mathcal{X} \subseteq \mathcal{H}$ but $x \notin \mathcal{H}$.

The idea behind analytic center cutting plane methods is to maintain a tractable outer approximation of the optimal set of (2). Then, in every iteration k, one approximates the analytic center x_k of this set. One of two things will happen:

- x_k does not lie in $\mathcal{X}$. In this case, use a separating hyperplane to remove x_k from the outer approximation of the feasible set.
- x_k lies in $\mathcal{X}$. Since x_k is feasible for (2), any optimal solution must have an objective value that is at least as good as $\langle c, x_k \rangle$. Any optimal solution will therefore lie in the halfspace $\{x \in \mathbb{R}^n : \langle c, x \rangle \leq \langle c, x_k \rangle\}$, and one may thus restrict the outer approximation to this halfspace.

The constraint $\|x\|^2 \leq 1$ from (1) will be included in our outer approximation explicitly. Hence, there are three remaining questions we have to answer before we can solve (1):

1. How will we generate separating hyperplanes for the constraint $\operatorname{mat}(x) \in \mathcal{COP}^d$?

2. How do we compute the analytic center of the outer approximation?

3. How can one prune constraints that do not have a large influence on the location of the analytic center?

These three questions will be answered in Sections 2.1, 2.2, and 2.3, respectively. Then, we state our algorithm in Section 2.4 and make some remarks concerning its complexity in Section 2.5.

2.1 Generating Cuts

The first question we will answer is how to generate separating hyperplanes for the copositive cone. Note that $X \in \mathbb{S}^d$ is copositive if and only if

$$\min_y \{y^\top X y : e^\top y = 1, y \geq 0\}, \tag{3}$$

where e is the all-ones vector, is nonnegative. It was shown by Xia, Vera, and Zuluaga [40] that the value of (3) is equal to the optimal value of the following mixed-integer linear program:

$$\left.\begin{array}{lll} \min\limits_{y,z,\mu,\nu} & -\mu & \\ \text{subject to} & Xy + \mu e - \nu = 0 & \\ & e^\top y = 1 & \\ & 0 \leq y_i \leq z_i & \forall i = 1, ..., d \\ & 0 \leq \nu_i \leq 2d(1 - z_i) \max\limits_{k,l} |X_{kl}| & \forall i = 1, ..., d \\ & z_i \in \{0, 1\} & \forall i = 1, ..., d, \end{array}\right\} \tag{4}$$

and that any optimal y from (4) is also an optimal solution for (3). If the optimal value of (4) is nonnegative, then X is copositive. If the optimal value of (4) is negative, then an optimal solution $y \geq 0$ from (4) admits the halfspace $\mathcal{H} = \{X' \in \mathbb{S}^d : y^\top X' y \geq 0\}$ such that $\mathcal{COP}^d \subset \mathcal{H}$ but $X \notin \mathcal{H}$. Note that this method was also used in Badenbroek and De Klerk [3].

As noted in Section 1, there are alternative methods to test matrix copositivity. Gaddum's method [19] is already outperformed by the above method for the 6×6 matrices in our test set, and our MILP method scales considerably better. The method by Nie et al. [34] can also become too slow for our purposes at moderate matrix dimensions. Anstreicher's recent method [1] also solves an MILP, which we expect to perform similar to (4).

In theory, we can therefore determine if a matrix is copositive by solving one MILP. In practice however, a solver may return a solution $(\hat{y}, \hat{z}, \hat{\mu}, \hat{\nu})$ to (4) where $\hat{y}^\top \hat{\nu} > 0$, violating the complementarity condition. This is caused by numerical tolerances allowing a solution with $\hat{z} \notin \{0, 1\}^d$, which mostly seems to occur if X has low rank (or is close to a low rank matrix). To find the optimal solution if this occurs, we fix z to the element-wise rounded value $\textsc{Round}(\hat{z})$ of $\hat{z}$. If the resulting problem is still feasible, we can compare its complementary solution with the solution to the model for $z \in \{0, 1\}^d \setminus \{\textsc{Round}(\hat{z})\}$. If the constraint $z = \textsc{Round}(\hat{z})$ does make the problem infeasible, we know that any optimal solution will have $z \in \{0, 1\}^d \setminus \{\textsc{Round}(\hat{z})\}$. The details of this procedure are given in Algorithm 1, where $\text{val}(\mathcal{M})$ denotes the objective value of the optimal solution returned by the solver when solving the model $\mathcal{M}$.

2.2 Approximating the Analytic Center

Now that we saw how to generate cuts for the copositive cone, we turn our attention to the second question: how to approximate the analytic center of our outer approximation. For the sake of concreteness, let us suppose the convex body $\mathcal{Q}$ for which we want to approximate the analytic center is the intersection of a ball and a polyhedron, i.e.

$$\mathcal{Q} = \left\{x \in \mathbb{R}^n : \|x\|^2 \leq r^2, \;\; a_i^\top x \leq b_i \, \forall i = 1, ..., m\right\}, \tag{5}$$

Algorithm 1 Method for testing copositivity or finding deep cuts

Input: Matrix $X \in \mathbb{S}^d$ which we want to test for copositivity.

1: **function** TestCopositive(X)
2: Let $\mathcal{M}$ refer to the model (4) with input X
3: $(\hat{y}, \hat{z}, \hat{\mu}, \hat{\nu}) \leftarrow$ SolveModel($\mathcal{M}$) ▷ See Line 10
4: **if** $\hat{y}^\top X \hat{y} \geq 0$ **then**
5: **return** true ▷ Returns true if X is copositive
6: **else** ▷ Returns a deep cut if X is not copositive:
7: **return** $\{\widehat{X} \in \mathbb{S}^d : \hat{y}^\top \widehat{X} \hat{y} \geq 0\}$ ▷ A halfspace $\mathcal{H}$ such that $\mathcal{COP}^d \subseteq \mathcal{H}$ but $X \notin \mathcal{H}$
8: **end if**
9: **end function**
10: **function** SolveModel($\mathcal{M}$, $u = +\infty$)
11: Let $(\hat{y}, \hat{z}, \hat{\mu}, \hat{\nu})$ be the solution to the model $\mathcal{M}$ returned by the solver
12: **if** $\hat{y}^\top \hat{\nu} > 0$ and $\text{val}(\mathcal{M}) < u$ **then**
13: Let $\overline{\mathcal{M}}$ be the model $\mathcal{M}$ with the added constraint $z = \text{Round}(\hat{z})$
14: Let $\mathcal{M}'$ be the model $\mathcal{M}$ with the added constraint $\sum_{i:\text{Round}(\hat{z}_i)=0} z_i + \sum_{i:\text{Round}(\hat{z}_i)=1} (1 - z_i) \geq 1$
15: **if** $\overline{\mathcal{M}}$ is feasible **then**
16: Compute the optimal solution to $\overline{\mathcal{M}}$, and SolveModel($\mathcal{M}', \text{val}(\overline{\mathcal{M}})$) if $\mathcal{M}'$ is feasible
17: **return** the solution with the best objective value out of these two
18: **else**
19: **return** SolveModel($\mathcal{M}'$)
20: **end if**
21: **else**
22: **return** the solution $(\hat{y}, \hat{z}, \hat{\mu}, \hat{\nu})$
23: **end if**
24: **end function**

where $a_1^\top, ..., a_m^\top$ are the rows of a matrix A, and $b \coloneqq (b_1, ..., b_m)$. The analytic center of $\mathcal{Q}$ is the optimal solution x to the problem

$$\inf_x \left\{ -\log(r^2 - \|x\|^2) - \sum_{i=1}^m \log(b_i - a_i^\top x) \right\}. \tag{6}$$

It is well known that self-concordant barrier functions only have an analytic center when their domain is bounded (see e.g. Renegar [37, Corollary 2.3.6]). This is why we use the upper bound r on $\|x\|$. Of course, a more traditional solution would be to ensure that the linear constraints $Ax \leq b$ describe a bounded set. We decided against this for reasons of numerical stability (more details in Section 2.5).

Since the objective function in (6) can only be evaluated at x where $\|x\|^2 < r^2$ and $Ax < b$, we will use an infeasible-start Newton method to solve (6). Similar to Boyd et al. [8, Section 2], one can reformulate the problem of computing the analytic center of $\mathcal{Q}$ as

$$\inf_{x,d,s} \left\{ -\log(d) - \sum_{i=1}^m \log(s_i) : d \leq r^2 - \|x\|^2, s \leq b - Ax \right\}, \tag{7}$$

which has Lagrangian

$$L(x, d, s, \kappa, \lambda) = -\log(d) - \sum_{i=1}^{m} \log(s_i) + \kappa(d - r^2 + \|x\|^2) + \lambda^\top (s - b + Ax),$$

with gradient

$$\nabla L(x, d, s, \kappa, \lambda) = \begin{bmatrix} 2\kappa x + A^\top \lambda \\ -d^{-1} + \kappa \\ -s^{-1} + \lambda \\ d - r^2 + \|x\|^2 \\ s - b + Ax \end{bmatrix}. \tag{8}$$

For the sake of completeness, let us show that it suffices to compute a stationary point of the Lagrangian.

Proposition 1. *Let $A \in \mathbb{R}^{m \times n}$ have rows $a_1^\top, ..., a_m^\top$, and let $b \in \mathbb{R}^m$, and $r > 0$. Let $\mathcal{Q}$ be as defined in (5), and assume that it has nonempty interior. Then, x_* is the analytic center of $\mathcal{Q}$ if and only if there exist $d_*, s_*, \kappa_*, \lambda_* > 0$ such that $\nabla L(x_*, d_*, s_*, \kappa_*, \lambda_*) = 0$.*

Proof. Because $\mathcal{Q}$ is nonempty and bounded, it has an analytic center, and problem (7) has an optimal solution. Since (7) is convex, a feasible solution (x_*, d_*, s_*) is optimal if and only if it satisfies the KKT conditions: there should exist κ_*, λ_* such that

$$\begin{cases} \begin{bmatrix} 2\kappa_* x_* + A^\top \lambda_* \\ -d_*^{-1} + \kappa_* \\ -s_*^{-1} + \lambda_* \end{bmatrix} = 0 \\ \kappa_*(d_* - r^2 + \|x_*\|^2) = 0 \\ \lambda_*^\top (s_* - b + Ax_*) = 0 \\ d_* \le r^2 - \|x_*\|^2 \\ s_* \le b - Ax_* \\ \kappa_*, \lambda_* \ge 0. \end{cases}$$

Since $\kappa_* = d_*^{-1} > 0$ and $\lambda_* = s_*^{-1} > 0$, the claim follows. □

The first order approximation for the Lagrangian shows

$$\nabla L(x + \Delta x, d + \Delta d, s + \Delta s, \kappa + \Delta\kappa, \lambda + \Delta\lambda) \approx \nabla L(x, d, s, \kappa, \lambda) + \nabla^2 L(x, d, s, \kappa, \lambda) \begin{bmatrix} \Delta x \\ \Delta d \\ \Delta s \\ \Delta\kappa \\ \Delta\lambda \end{bmatrix}, \tag{9}$$

which means we can solve a linear system to find the Newton step $(\Delta x, \Delta d, \Delta s, \Delta\kappa, \Delta\lambda)$ with which we can approximate a stationary point of L. Next, we find that

$$\nabla^2 L(x, d, s, \kappa, \lambda) = \begin{bmatrix} 2\kappa I & 0 & 0 & 2x & A^\top \\ 0 & d^{-2} & 0 & 1 & 0 \\ 0 & 0 & \mathrm{Diag}(s^{-2}) & 0 & I \\ 2x^\top & 1 & 0 & 0 & 0 \\ A & 0 & I & 0 & 0 \end{bmatrix}. \tag{10}$$

Thus, if we substitute the expressions (8) and (10) in (9), we see that the Newton step should satisfy

$$0 = \begin{bmatrix} 2\kappa x + A^\top \lambda \\ -d^{-1} + \kappa \\ -s^{-1} + \lambda \\ d - r^2 + \|x\|^2 \\ s - b + Ax \end{bmatrix} + \begin{bmatrix} 2\kappa I & 0 & 0 & 2x & A^\top \\ 0 & d^{-2} & 0 & 1 & 0 \\ 0 & 0 & \mathrm{Diag}(s^{-2}) & 0 & I \\ 2x^\top & 1 & 0 & 0 & 0 \\ A & 0 & I & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta d \\ \Delta s \\ \Delta\kappa \\ \Delta\lambda \end{bmatrix}. \tag{11}$$

We could solve this system directly, but it is more efficient to note that the last conditions imply

$$\begin{cases} \Delta\kappa &= -\kappa + d^{-1} - d^{-2}\Delta d \\ \Delta\lambda &= -\lambda + s^{-1} - \text{Diag}(s^{-2})\Delta s \\ \Delta d &= -d + r^2 - \|x\|^2 - 2x^\top \Delta x \\ \Delta s &= -s + b - Ax - A\Delta x \end{cases} \tag{12}$$

which means the entire Newton step can be expressed in terms of Δx. The first n equations of the Newton system (11) are thus

$$\begin{aligned} -2\kappa x - A^\top \lambda &= 2\kappa\Delta x + 2x\Delta\kappa + A^\top \Delta\lambda \\ &= 2\kappa\Delta x + 2x[d^{-1} - \kappa - d^{-2}\Delta d] + A^\top[s^{-1} - \lambda - \text{Diag}(s^{-2})\Delta s] \\ &= 2\kappa\Delta x + 2x[d^{-1} - \kappa - d^{-2}(-d + r^2 - \|x\|^2 - 2x^\top \Delta x)] \\ &\quad + A^\top[s^{-1} - \lambda - \text{Diag}(s^{-2})(-s + b - Ax - A\Delta x)] \end{aligned}$$

or equivalently,

$$\left[2\kappa I + \frac{4}{d^2}xx^\top + A^\top \text{Diag}(s^{-2})A\right]\Delta x = \frac{r^2 - \|x\|^2 - 2d}{d^2}2x + A^\top \text{Diag}(s^{-2})(b - Ax - 2s). \tag{13}$$

After solving this system for Δx, we can compute the other components of the Newton step through equations (12). Now that it is clear how one can compute the Newton step for problem (7), we propose Algorithm 2 to solve (7).

Let us make a few observations about this algorithm. First, note that if $\kappa > 0$, the matrix $2\kappa I + 4d^{-2}xx^\top + A^\top \text{Diag}(s^{-2})A$ is positive definite, and hence invertible. Thus, as long as $\kappa > 0$, the system (13) will have a (unique) solution Δx.

Second, the value for t in Line 10 of Algorithm 2 is chosen such that after the update, d, s, and κ will all remain positive. In principle, the value 0.9 could be replaced by any real number from $(0, 1)$. Note that we are not requiring that λ remains positive in all iterations: numerical evidence suggests that the method is more likely to succeed if some elements of λ are allowed to be negative in some iterations. Nevertheless, Algorithm 2 only returns a success status if the final λ is nonnegative.

Third, the algorithm returns the current solution x with success status in two cases. In either case, the current solution should approximately be a stationary point of the Lagrangian, i.e. the norm of $\nabla L(x, d, s, \kappa, \lambda)$ has to be small, and we should have $\lambda \geq 0$. Moreover, one of the following conditions should hold:

1. Updating the point by adding t times the Newton step leads to a larger norm of the Lagrangian gradient. In this case, taking the step does not improve the solution. Since the current point is already approximately a stationary point, this solution is returned;

2. We are in iteration $k_{\max}$. Since the current point is approximately a stationary point, this solution is returned.

The reason to continue taking Newton steps even if the norm of the Lagrangian's gradient is small is that Newton's method converges very rapidly when the current point is near the optimum. By running just a few more iterations, we get a solution with much higher accuracy.

Finally, compared to the algorithm in Boyd et al. [8, Section 2], Algorithm 2 does not use backtracking line search. The reason is that for problem (7), the norm of the Lagrangian gradient does not seem to decrease monotonically during the algorithm's run. In fact, the norm of this gradient usually first decreases to the order 10^0, then increases slightly to the order 10^1, before decreasing rapidly to the order 10^{-8}. If one does backtracking line search on t to ensure that in every iteration the norm of the gradient decreases, the values of t can become very small (say, of the order 10^{-9}). Then, the number of iterations required to achieve convergence would be impractically large.

Algorithm 2 Infeasible start Newton method for (7)

Input: Convex body $\mathcal{Q} = \{x \in \mathbb{R}^n : \|x\|^2 \leq r^2, Ax \leq b\}$, where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$; starting point $x_0 \in \mathbb{R}^n$; maximum number of iterations $k_{\max} = 50$; gradient norm tolerance $\delta = 10^{-8}$.

function AnalyticCenter($\mathcal{Q}, x_0$)
  $k \leftarrow 1$
  $d_0 \leftarrow \begin{cases} r^2 - \|x_0\|^2 & \text{if } r^2 - \|x\|^2 > 0 \\ 1 & \text{otherwise} \end{cases}$
  $(s_0)_i \leftarrow \begin{cases} b_i - a_i^\top x & \text{if } b_i - a_i^\top x > 0 \\ 1 & \text{otherwise} \end{cases}$ for all $i = 1, ..., m$
  $\kappa_0 \leftarrow -1$
  $\lambda_0 \leftarrow 0$
  while $k \leq k_{\max}$ **do**
    Compute Δx_k from (13)
    Compute $(\Delta d_k, \Delta s_k, \Delta\kappa_k, \Delta\lambda_k)$ from (12)
    $t_k \leftarrow \min\{1, 0.9 \times \sup\{t \geq 0 : d_k + t\Delta d_k \geq 0, s_k + t\Delta s_k \geq 0, \kappa_k + t\Delta\kappa_k \geq 0\}\}$
    $g_k(t) := \|\nabla L(x_k + t\Delta x_k, d_k + t\Delta d_k, s_k + t\Delta s_k, \kappa_k + t\Delta\kappa_k, \lambda_k + t\Delta\lambda_k)\|$
    if $g_k(0) \leq \delta$ and $\lambda_k \geq 0$ and ($g_k(t_k) \geq g_k(0)$ or $k = k_{\max}$) **then**
      return x_k with success status
    end if
    $k \leftarrow k + 1$
  end while
  return x_k with failure status
end function

2.3 Pruning Constraints

The next question we should answer is how we can prune constraints from our outer approximation (5). Pruning is often used to reduce the number of constraints defining the outer approximation, which means keeps the computational effort per iteration stable. Moreover, the linear system (13) will quickly become ill-conditioned if no constraints are dropped.

The idea we use is the same as in Boyd, Vandenberghe, and Skaf [8, Section 3]: denote the barrier of which we compute the analytic center by

$$\Phi(x) := -\log(r^2 - \|x\|^2) - \sum_{i=1}^{m} \log(b_i - a_i^\top x). \tag{14}$$

Since Φ is self-concordant, the Dikin ellipsoid around the analytic center of Φ is contained in $\mathcal{Q}$, i.e.

$$\{x \in \mathbb{R}^n : (x - x_*)^\top \nabla^2\Phi(x_*)(x - x_*) \leq 1\} \subseteq \mathcal{Q}, \tag{15}$$

where x_* is the minimizer of Φ and

$$\nabla^2\Phi(x_*) = \frac{2}{r^2 - \|x_*\|^2} I + \frac{4}{(r^2 - \|x_*\|^2)^2} x_* x_*^\top + \sum_{i=1}^{m} \frac{1}{(b_i - a_i^\top x_*)^2} a_i a_i^\top.$$

Moreover, it will be shown at the end of this section that for our outer approximation $\mathcal{Q}$ it holds that

$$\mathcal{Q} \subseteq \{x \in \mathbb{R}^n : (x - x_*)^\top \nabla^2\Phi(x_*)(x - x_*) \leq (m + 1)^2\}. \tag{16}$$

Hence, following [8], we define the relevance measure

$$\eta_i \coloneqq \frac{b_i - a_i^\top x_*}{\sqrt{a_i^\top \nabla^2 \Phi(x_*)^{-1} a_i}}, \tag{17}$$

for all linear constraints $i = 1, ..., m$. By (15), all η_i are at least one. Moreover, it follows from (16) that if $\eta_i \geq m + 1$, the corresponding constraint is certainly redundant.

With this in mind, we propose Algorithm 3 to prune constraints from $\mathcal{Q}$. Note that the ball constraint $\|x\|^2 \leq r^2$ is never pruned.

Algorithm 3 A pruning method for the intersection of a ball and a polyhedron

Input: Convex body $\mathcal{Q} = \{x \in \mathbb{R}^n : \|x\|^2 \leq r^2, Ax \leq b\}$, where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$; analytic center x_* of $\mathcal{Q}$; maximum number of linear inequalities $m_{\max} = 3n$.

function PRUNE($\mathcal{Q}, x_*$)
  if $m > n$ **then**
    Compute η_i as in (17) for $i = 1, ..., m$
    Remove all constraints $a_i^\top x \leq b_i$ with $\eta_i \geq m + 1$ from $\mathcal{Q}$
    if $\mathcal{Q}$ still contains more than $m_{\max}$ linear inequalities **then**
      Remove the constraints $a_i^\top x \leq b_i$ with the largest values of η_i from $\mathcal{Q}$ such that $m_{\max}$ remain
    end if
  end if
  return $\mathcal{Q}$
end function

As an alternative, one might consider dropping $m - m_{\max}$ constraints, possibly keeping some redundant constraints. The reason we do not adopt this strategy is that we noticed Algorithm 3 leads to slightly better numerical performance on our test sets.

We finish this section with a proof of the relation (16).

Proposition 2. *Let $A \in \mathbb{R}^{m \times n}$ have rows $a_1^\top, ..., a_m^\top$, and let $b \in \mathbb{R}^m$, and $r > 0$. Let $\mathcal{Q}$ be as defined in (5), and assume that it has nonempty interior. Define Φ as in (14), and let x_* be the minimizer of Φ. Then, for any $x \in \operatorname{dom} \Phi$, we have*

$$(x - x_*)^\top \nabla^2 \Phi(x_*)(x - x_*) \leq (m + 1)^2.$$

Proof. Define the barrier function

$$f(t, \tilde{x}, s) \coloneqq -\log(t^2 - \|\tilde{x}\|^2) - \sum_{i=1}^m \log(s_i),$$

whose domain is a symmetric cone. The barrier parameter ϑ of f satisfies $\vartheta \leq m + 1$. Note that any $x \in \operatorname{dom} \Phi$ if and only if $(r, x, b - Ax) \in \operatorname{dom} f$. We will first show that the gradient of f at $(r, x_*, b - Ax_*)$ is orthogonal to $(r, x, b - Ax) - (r, x_*, b - Ax_*) = (0, x - x_*, A(x_* - x))$. The claim will then follow from a property of symmetric cones.

The gradient of f is

$$\nabla f(t, \tilde{x}, s) \coloneqq \begin{bmatrix} -2t/(t^2 - \|\tilde{x}\|^2) \\ 2x/(t^2 - \|\tilde{x}\|^2) \\ -s^{-1} \end{bmatrix},$$

so it follows that

$$\nabla f(r, x_*, b - Ax_*)^\top \begin{bmatrix} 0 \\ x - x_* \\ A(x_* - x) \end{bmatrix} = \frac{2x_*^\top (x - x_*)}{r^2 - \|x_*\|^2} - \sum_{i=1}^m \frac{a_i^\top (x_* - x)}{b_i - a_i^\top x_*}. \tag{18}$$

Because x_* is the minimizer of the convex function Φ, we have

$$0 = \nabla\Phi(x_*) = \frac{2}{r^2 - \|x_*\|^2}x_* + \sum_{i=1}^{m} \frac{1}{b_i - a_i^\top x_*}a_i,$$

which implies that (18) is zero. Therefore, Theorem 3.5.9 in Renegar [37] shows that

$$\begin{bmatrix} 0 \\ x - x_* \\ A(x_* - x) \end{bmatrix}^\top \nabla^2 f(r, x_*, b - Ax_*) \begin{bmatrix} 0 \\ x - x_* \\ A(x_* - x) \end{bmatrix} \leq \vartheta^2, \tag{19}$$

where $\nabla^2 f$ is the Hessian of f, given by

$$\nabla^2 f(t, \tilde{x}, s) \coloneqq \frac{1}{(t^2 - \|\tilde{x}\|^2)^2} \begin{bmatrix} 2t^2 + \|\tilde{x}\|^2 & -4t\tilde{x}^\top & 0 \\ -4t\tilde{x}^\top & 2(t^2 - \|\tilde{x}\|^2)I + 4\tilde{x}\tilde{x}^\top & 0 \\ 0 & 0 & (t^2 - \|\tilde{x}\|^2)^2 \operatorname{Diag}(s^{-2}) \end{bmatrix}.$$

In other words, (19) is equivalent to

$$(x - x_*)^\top \left[\frac{2}{r^2 - \|x_*\|^2}I + \frac{4}{(r^2 - \|x_*\|^2)^2}x_*x_*^\top\right](x - x_*) + \sum_{i=1}^{m} \frac{(a_i^\top(x_* - x))^2}{(b_i - a_i^\top x_*)^2} \leq \vartheta^2,$$

which proves the claim, since $\vartheta \leq m + 1$. □

2.4 Algorithm Description

Now that we answered the major questions surrounding an ACCP method for checking complete positivity of a matrix, we more on to our final method. We start with a quite general analytic center cutting plane method, and then add a wrapper function that performs the complete positivity check. The reason for making this split is that it makes our code easy to extend when solving other copositive optimization problems for which a bound on the norm of an optimal solution is known. We state our proposed analytic center cutting plane method to solve (2) in Algorithm 4.

We continue the algorithm even if we cannot find the analytic center to high accuracy. Late in the algorithm's run, the system (13) often becomes ill-conditioned. This is to be expected, since as Algorithm 4 progresses, the outer approximation $\mathcal{Q}_k$ becomes smaller and smaller. The distance from the analytic center to the linear constraints also goes to zero, but not at the same pace for every constraint. We may arrive in a situation where $b_i - a_i^\top x_k$ is of the order 10^{-4} for some constraints i, and of the order 10^{-8} for other constraints. This causes a considerable spread in the eigenvalues of the matrix in (13).

If the analytic center is not known to a decent accuracy, the pruning procedure in Algorithm 3 may remove constraints that are actually very important to the definition of $\mathcal{Q}_k$. One could of course still run the pruning function using the inaccurate analytic center approximation. However, because the problems in the analytic center computation only occur late in the algorithm's run, pruning or not pruning with the inaccurate approximation does not seem to have a major impact on total runtime.

Algorithm 4 is a (relatively) general analytic center cutting plane method. The problem (1) can be solved by calling Algorithm 4 with the right parameters, as is done by Algorithm 5.

2.5 A Note on Complexity

Our aim in this paper is to propose an algorithm with good practical performance. This is why we placed emphasis on a robust copositivity check, constraint pruning, and efficient computation of the analytic center. However, such an algorithm does not lend itself well to a formal complexity analysis. For instance, to the best of the authors' knowledge, the only analysis in the literature of an analytic center cutting plane method with

Algorithm 4 Analytic Center Cutting Plane method to solve (2)

Input: Objective $c \in \mathbb{R}^n$; oracle function $\textsc{Oracle} : \mathbb{R}^n \to \{\text{true}\} \cup \{\{x \in \mathbb{R}^n : a^\top x \leq b\} : a \in \mathbb{R}^n, b \in \mathbb{R}\}$; radius $r > 0$; optimality tolerance $\epsilon = 10^{-6}$.

1: **function** ACCP(c, Oracle, r)
2: $\quad\mathcal{Q}_1 \leftarrow \{x \in \mathbb{R}^n : \|x\|^2 \leq r^2\}$
3: $\quad x_0 \leftarrow 0$
4: $\quad k \leftarrow 1$
5: $\quad$**while** the best feasible solution so far x_* has $\textsc{RelativeGap}(c, x_*, \mathcal{Q}_k) > \epsilon$ **do** ▷ See Line 22
6: $\quad\quad x_k \leftarrow \textsc{AnalyticCenter}(\mathcal{Q}_k, x_{k-1})$
7: $\quad\quad$**if** AnalyticCenter terminated with a failure status **then**
8: $\quad\quad\quad$Check if $x_k \in \operatorname{int} \mathcal{Q}_k$. If not, throw an error.
9: $\quad\quad$**else**
10: $\quad\quad\quad\mathcal{Q}_k \leftarrow \textsc{Prune}(\mathcal{Q}_k, x_k)$
11: $\quad\quad$**end if**
12: $\quad\quad$**if** $\textsc{Oracle}(x_k)$ returns true **then**
13: $\quad\quad\quad\mathcal{Q}_{k+1} \leftarrow \mathcal{Q}_k \cap \{x \in \mathbb{R}^n : c^\top x/\|c\| \leq c^\top x_k/\|c\|\}$
14: $\quad\quad$**else** ▷ $\textsc{Oracle}(x_k)$ returns a halfspace
15: $\quad\quad\quad\mathcal{H}_k = \{x \in \mathbb{R}^n : a_k^\top x \leq b_k\}$ is the halfspace returned by $\textsc{Oracle}(x_k)$
16: $\quad\quad\quad\mathcal{Q}_{k+1} \leftarrow \mathcal{Q}_k \cap \{x \in \mathbb{R}^n : a_k^\top x/\|a_k\| \leq b_k/\|a_k\|\}$
17: $\quad\quad$**end if**
18: $\quad\quad k \leftarrow k + 1$
19: $\quad$**end while**
20: $\quad$**return** the best feasible solution found x_*
21: **end function**
22: **function** $\textsc{RelativeGap}(c, x_*, \mathcal{Q})$
23: $\quad l \leftarrow \min_x\{c^\top x : x \in \mathcal{Q}\}$
24: $\quad$**return** $(c^\top x_* - l)/(1 + \min\{|c^\top x_*|, |l|\})$
25: **end function**

Algorithm 5 A wrapper function to determine if a matrix is completely positive by solving (1)

Input: $C \in \mathbb{S}^d$ for which we want to determine if $C \in \mathcal{CP}^d$ or not.

1: **function** CompletelyPositiveCut(C)
2: $\quad c \leftarrow \operatorname{mat}^*(C)$
3: $\quad r \leftarrow 1$
4: $\quad\textsc{Oracle}(x) \leftarrow \textsc{TestCopositive}(\operatorname{mat}(x))$
5: $\quad$**return** $\operatorname{mat}(\text{ACCP}(c, \textsc{Oracle}, r))$
6: **end function**

constraint pruning is due to Atkinson and Vaidya [2]. Although the number of constraints in their algorithm is technically bounded by a polynomial of n, this bound is so large as to be uninteresting in practice.

The analysis that perhaps comes closest to covering our algorithm is the survey by Goffin and Vial [22], who find a polynomial number of iterations for an analytic center cutting plane method with deep cuts.

Their method only uses linear constraints, and does not prune cuts. Moreover, the method of recovering a feasible solution after adding a deep cut is different from the infeasible start Newton method we use.

Nevertheless, we compared our method numerically to Goffin and Vial's, and found that our method exhibits somewhat better numerical performance on our test set. In particular, Goffin and Vial's method struggles earlier to approximate the analytic center. Whereas we could solve the problems in our test set up to a relative gap of 10^{-6}, Goffin and Vial's method sometimes failed to recover a point in the feasible set when the relative gap was still of the order 10^{-5}. The condition number of their linear systems had become very large at this point, explaining the inaccuracy. At this level of the relative gap, the condition number of the system (13) in our algorithm was somewhat lower.

In short, while our method is not covered by a formal complexity analysis, we do prefer it over other algorithms in the literature for numerical reasons.

3 Numerical Experiments

3.1 Extremal Matrices of the 6×6 Doubly Nonnegative Cone

We test Algorithm 5 on extremal matrices from the doubly nonnegative cone. Ten of such 6×6 matrices were proposed in [3, Appendix B]. We run Algorithm 5 on these matrices, and record the number of calls to TestCopositive. For the sake of comparison, we also applied the ellipsoid method of Yudin and Nemirovski [43]. The termination criterion for the Ellipsoid method is similar to that in Algorithm 4, i.e. the relative gap can be at most 10^{-6}. The only difference is that in the case of the Ellipsoid method, the lower bound is computed through minimization over the current ellipsoid, not over some outer approximation $\mathcal{Q}$.

The results are shown in Table 1. We record the final objective value for all instances and both methods, as well as the number of calls to TestCopositive. The reason to report this number of calls is that the oracle performs the theoretically intractable part of these methods: testing if a matrix is copositive. All other parts of the ellipsoid method or Algorithm 4 complete in polynomial time for each oracle call. Hence, to get the best performance for larger matrices, one would like to minimize the number of oracle calls.

	Final objective value		TestCopositive calls	
Name	Algorithm 4	Ellipsoid method	Algorithm 4	Ellipsoid method
`extremal_rand_1`	-0.28140	-0.28139	169	8560
`extremal_rand_2`	-0.72121	-0.72123	166	8030
`extremal_rand_3`	-0.73676	-0.73676	163	8598
`extremal_rand_4`	-0.54867	-0.54867	164	7910
`extremal_rand_5`	-0.92462	-0.92460	177	8546
`extremal_rand_6`	-1.42946	-1.42946	168	8184
`extremal_rand_7`	-1.67891	-1.67889	168	9119
`extremal_rand_8`	-1.24450	-1.24450	165	8126
`extremal_rand_9`	-1.04975	-1.04974	176	8318
`extremal_rand_10`	-0.68582	-0.68583	167	7950

Table 1: Objective values returned by Algorithm 5 and by the Ellipsoid method, applied to the matrices from [3, Appendix B].

As can be seen from Table 1, both methods manage to find deep cuts that separate the matrices from the completely positive cone. However, Algorithm 4 does this with roughly 50 times fewer calls to the copositivity oracle.

3.2 Matrices on the Boundary of the Doubly Nonnegative Cone in Higher Dimensions

To investigate how the algorithm scales, we also generated test instances in higher dimensions. To the best of our knowledge, a complete characterization of the extremal rays of the $d \times d$ doubly nonnegative cone is unknown for $d > 6$. (See the corollary to Theorem 3.1, and Propositions 5.1 and 6.1 in Ycart [41] for the extremal matrices for $d \leq 6$.) Hence, we use a semidefinite programming heuristic to find doubly nonnegative matrices in these dimensions which are not completely positive.

The matrices used in Section 3.1 are 6×6 doubly nonnegative matrices C with rank 3 and the entries $C_{i,i+1} = 0$ for all $i \in \{1, ..., 5\}$. This pattern of zeros can of course be extended to higher dimensions, but the low rank criterion is not tractable in semidefinite programming. The standard trick to find a low-rank solution – which we also adopt – is to minimize the trace of the matrix variable, see e.g. Fazel, Hindi, and Boyd [18] and the references therein. To create a $d \times d$ test instance, we thus run the procedure in Algorithm 6.

Algorithm 6 A heuristic procedure to generate random matrices on the boundary of the doubly nonnegative cone

Input: Dimension d of a random matrix $C \in \mathbb{S}^d$ to generate.

1: $R_0 \in \mathbb{R}^{d \times d}$ is a matrix whose elements are samples from a standard normal distribution

2: $R \leftarrow |R_0| + |R_0|^\top$, where $|R_0| = [|(R_0)_{ij}|]$ is the element-wise absolute value

3: Let C^* be an (approximately) optimal solution to

$$\begin{aligned} &\inf_C \operatorname{tr} C + \tfrac{1}{2} d \|C - R\| \\ &\text{subject to } C_{i,i+1} = 0 \qquad \forall i \in \{1, ..., d-1\} \\ &\qquad C \succeq 0, C \geq 0. \end{aligned}$$

4: **for** $j \in \{1, ..., 10\}$ **do**

5: Set all eigenvalues of C^* smaller than 10^{-6} to zero

6: Set all elements of C^* smaller than 10^{-4} to zero

7: **end for**

8: **return** $C^*/\|C^*\|$

The objective in Line 3 of Algorithm 6 includes two terms: the term $\operatorname{tr} C$ to get a low-rank solution, and the term $\frac{1}{2} d \|C - R\|$ to get a solution close to our random matrix R. Without this last term, the optimal solution of the problem would be the zero matrix. The weight $\frac{1}{2} d$ was chosen because numerical experiments suggested this weight leads to solutions with low rank, but not rank zero, for the dimensions in our test set. The solution C^* computed in Line 3 by interior point methods still lies in the interior of the doubly nonnegative cone. To project this solution to the boundary of the doubly nonnegative cone, we run the Lines 4 to 7.

For each $d \in \{6, 7, 8, 9, 10, 15, 20, 25\}$, we generated ten test instances with Algorithm 6. Such an instance C is only included in the final test set if Algorithm 5 returns an X such that $\langle C, X \rangle < -0.01$, which was almost always the case. In those few cases where $\langle C, X \rangle \geq -0.01$, a new instance was generated. Hence, we end up with ten $d \times d$ doubly nonnegative matrices that are not completely positive, for each $d \in \{6, 7, 8, 9, 10, 15, 20, 25\}$. These instances are available at `https://github.com/rileybadenbroek/CopositiveAnalyticCenter.jl/tree/master/test`.

Algorithm 5 is applied to each of these instances, and the total number of calls to TestCopositive is reported in Figure 1. (We do not report these results as in Table 1 since there are 80 instances, and running the ellipsoid method for all of them would take too much time.) As one can see, the number of oracle calls for one of our test instances with dimension d is roughly $7d^{5/3}$.

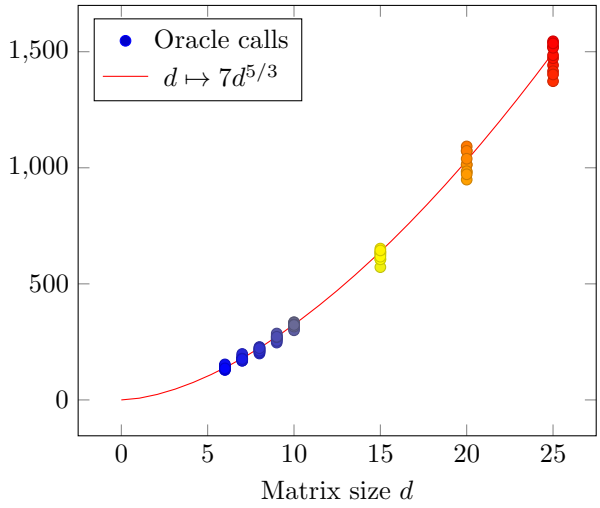

Figure 1: Number of oracle calls in Algorithm 5 for the $d \times d$ test instances generated with Algorithm 6

4 Conclusion

We have proposed an analytic center cutting plane algorithm to separate a matrix from the completely positive cone. This algorithm solves an optimization problem over the copositive cone, where membership of the copositive cone is tested through a mixed-integer linear program. We have emphasized stable numerical performance, which leads to an algorithm for which we do not have a formal complexity analysis. On the other hand, the numerical results are encouraging. In particular, the number of oracle calls to test matrix copositivity grows roughly like $O(d^2)$ for $d \times d$ matrices. Thus one can leverage the recent progress on testing matrix copositivity [3] or the similar method by Anstreicher [1]. We have therefore made some computational progress on an open problem formulated by Berman, Dür, and Shaked-Monderer [4]. It is worthwhile to note that our algorithm can be applied to any copositive optimization problem, as long as an upper bound on the norm of the optimal solution is known. The code is available at `https://github.com/rileybadenbroek/CopositiveAnalyticCenter.jl`.

References

[1] K. M. Anstreicher. Testing copositivity via mixed–integer linear programming. *Optimization Online preprint*, 2020. `http://www.optimization-online.org/DB_HTML/2020/03/7659.html`.

[2] D. S. Atkinson and P. M. Vaidya. A cutting plane algorithm for convex programming that uses analytic centers. *Mathematical Programming*, 69(1-3):1–43, 1995.

[3] R. Badenbroek and E. de Klerk. Simulated annealing with hit-and-run for convex optimization: rigorous complexity analysis and practical perspectives for copositive programming. *arXiv preprint arXiv:1907.02368*, 2019.

[4] A. Berman, M. Dur, and N. Shaked-Monderer. Open problems in the theory of completely positive and copositive matrices. *Electronic Journal of Linear Algebra*, 29(1):46–58, 2015.

[5] I. M. Bomze. Building a completely positive factorization. *Central European Journal of Operations Research*, 26(2):287–305, 2018.

[6] I. M. Bomze and E. De Klerk. Solving standard quadratic optimization problems via linear, semidefinite and copositive programming. *Journal of Global Optimization*, 24(2):163–185, 2002.

[7] I. M. Bomze, F. Jarre, and F. Rendl. Quadratic factorization heuristics for copositive programming. *Mathematical Programming Computation*, 3(1):37–57, 2011.

[8] S. Boyd, L. Vandenberghe, and J. Skaf. Lecture notes analytic center cutting-plane method. `https://pdfs.semanticscholar.org/368e/4045bf62a2bb3fb4a39f0b0b86d7ab295964.pdf`, April 2011.

[9] S. Bundfuss and M. Dür. Algorithmic copositivity detection by simplicial partition. *Linear Algebra and its Applications*, 428(7):1511–1523, 2008.

[10] S. Bundfuss and M. Dür. An adaptive linear approximation algorithm for copositive programs. *SIAM Journal on Optimization*, 20(1):30–53, 2009.

[11] S. Burer. On the copositive representation of binary and continuous nonconvex quadratic programs. *Mathematical Programming*, 120(2):479–495, 2009.

[12] S. Burer and H. Dong. Separation and relaxation for cones of quadratic forms. *Mathematical Programming*, 137(1-2):343–370, 2013.

[13] E. De Klerk and D. V. Pasechnik. Approximation of the stability number of a graph via copositive programming. *SIAM Journal on Optimization*, 12(4):875–892, 2002.

[14] P. J. Dickinson. A new certificate for copositivity. *Linear Algebra and its Applications*, 569:15–37, 2019.

[15] P. J. Dickinson and M. Dür. Linear-time complete positivity detection and decomposition of sparse matrices. *SIAM Journal on Matrix Analysis and Applications*, 33(3):701–720, 2012.

[16] P. J. Dickinson and L. Gijben. On the computational complexity of membership problems for the completely positive cone and its dual. *Computational Optimization and Applications*, 57(2):403–415, 2014.

[17] V. Elser. Matrix product constraints by projection methods. *Journal of Global Optimization*, 68(2):329–355, 2017.

[18] M. Fazel, H. Hindi, and S. P. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *Proceedings of the 2001 American Control Conference*, volume 6, pages 4734–4739. IEEE, 2001.

[19] J. W. Gaddum et al. Linear inequalities and quadratic forms. *Pacific Journal of Mathematics*, 8(3):411–414, 1958.

[20] J.-L. Goffin, Z.-Q. Luo, and Y. Ye. Complexity analysis of an interior cutting plane method for convex feasibility problems. *SIAM Journal on Optimization*, 6(3):638–652, 1996.

[21] J.-L. Goffin and J.-P. Vial. On the computation of weighted analytic centers and dual ellipsoids with the projective algorithm. *Mathematical Programming*, 60(1-3):81–92, 1993.

[22] J.-L. Goffin and J.-P. Vial. Convex nondifferentiable optimization: A survey focused on the analytic center cutting plane method. *Optimization Methods and Software*, 17(5):805–867, 2002.

[23] P. Groetzner and M. Dür. A factorization method for completely positive matrices. *Optimization Online*, 2018.

[24] J.-B. Hiriart-Urruty and A. Seeger. A variational approach to copositive matrices. *SIAM Review*, 52(4):593–629, 2010.

[25] F. Jarre and K. Schmallowsky. On the computation of C^* certificates. *Journal of Global Optimization*, 45(2):281, 2009.

[26] J. Kemperman and M. Skibinsky. Covariance spaces for measures on polyhedral sets. *Lecture Notes-Monograph Series*, pages 182–195, 1992.

[27] Q. Kong, C.-Y. Lee, C.-P. Teo, and Z. Zheng. Scheduling arrivals to a stochastic service delivery system using copositive cones. *Operations Research*, 61(3):711–726, 2013.

[28] J. B. Lasserre. New approximations for the cone of copositive matrices and its dual. *Mathematical Programming*, 144(1-2):265–276, 2014.

[29] X. Li, K. Natarajan, C.-P. Teo, and Z. Zheng. Distributionally robust mixed integer linear programs: Persistency models with applications. *European Journal of Operational Research*, 233(3):459–473, 2014.

[30] T. S. Motzkin and E. G. Straus. Maxima for graphs and a new proof of a theorem of turán. *Canadian Journal of Mathematics*, 17:533–540, 1965.

[31] K. G. Murty and S. N. Kabadi. Some NP-complete problems in quadratic and nonlinear programming. *Mathematical Programming*, 39(2):117–129, 1987.

[32] K. Natarajan, C. P. Teo, and Z. Zheng. Mixed 0-1 linear programs under objective uncertainty: A completely positive representation. *Operations Research*, 59(3):713–728, 2011.

[33] J. Nie. The $\mathcal{A}$-truncated K-moment problem. *Foundations of Computational Mathematics*, 14(6):1243–1276, 2014.

[34] J. Nie, Z. Yang, and X. Zhang. A complete semidefinite algorithm for detecting copositive matrices and tensors. *SIAM Journal on Optimization*, 28(4):2902–2921, 2018.

[35] P. A. Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. PhD thesis, California Institute of Technology, 2000.

[36] J. Peña, J. Vera, and L. F. Zuluaga. Computing the stability number of a graph via linear and semidefinite programming. *SIAM Journal on Optimization*, 18(1):87–105, 2007.

[37] J. Renegar. *A Mathematical View of Interior-Point Methods in Convex Optimization*. SIAM, 2001.

[38] M. D. Sikirić, A. Schürmann, and F. Vallentin. A simplex algorithm for rational cp-factorization. *Mathematical Programming*, 2020. `https://doi.org/10.1007/s10107-020-01467-4`.

[39] J. Sponsel and M. Dür. Factorization and cutting planes for completely positive matrices by copositive projection. *Mathematical Programming*, 143(1-2):211–229, 2014.

[40] W. Xia, J. C. Vera, and L. F. Zuluaga. Globally solving non-convex quadratic programs via linear integer programming techniques. *INFORMS Journal on Computing*, 2018.

[41] B. Ycart. Extrémales du cône des matrices de type non négatif, à coefficients positifs ou nuls. *Linear Algebra and its Applications*, 48:317–330, 1982.

[42] E. A. Yıldırım. On the accuracy of uniform polyhedral approximations of the copositive cone. *Optimization Methods and Software*, 27(1):155–173, 2012.

[43] D. Yudin and A. Nemirovski. Informational complexity and effective methods of solution of convex extremal problems. *Economics and Mathematical Methods*, 12:357–369, 1976.